ANDROID MOBILE PLATFORM SECURITY AND MALWARE SURVEY

Ameya Nayak[1], Tomas Prieto[2], Mohammad Alshamlan[3], Kang Yen[4]

[1,2,3,4] *Electrical and Computer Engineering Department, Florida International University, Miami, FL, USA*

Abstract

As mobile devices become ubiquitous, more people and companies are readily adopting the technology to conduct day-to-day business, and are increasing the amount of personal data transmitted and stored on these devices. These devices are now part of a global infrastructure powering communication and how we do business around the world. In turn, the inherent vulnerabilities are becoming an ever more critical topic of interest and challenge as we continue to see a rapid rate of malware development. This paper is a comprehensive survey on a broad view of the growing Android community, its rapidly growing malware attacks, and security concerns. Serving to aid in the continuous challenge of identifying current and future vulnerabilities as well as incorporating security strategies against them, this survey will focus primarily on mobile devices (also known as smart phones) running the Android mobile operating system between the years of 2007 and 2013.

Index Terms: *mobile, Android, malware, security*

----------***----------

1. INTRODUCTION

The fast growth and large popularity of Android powered mobile devices has raised a lot of concerns when it comes to security. In 2013 it was reported that an approximate of 550,000 Android powered phones were activated on a daily basis [23]. Since its inception, Android was meant to unify mobile users with common interfaces, applications, Application Programming Interfaces (API's), simplifying the development for a broad audience. With such a far-reaching platform**,** the arena for attacks from malware developer has become more accessible and increasingly enticing. A constantly and rapidly evolving collection of malware is entering the mobile software space as thousands of malware instances are being detected quarterly [22].

People use smart phones for shopping, banking, e-mail, and other activities that require passwords and payment information. Banks rely on cell phones for two-factor authentication. Users may also save authentication and payment credentials in text documents on their phones (for example, to use the phone as a mobile password manager). This makes cell phones a target for credential theft. As of 2008, bank account credentials, credit card numbers, and e-mail account passwords were worth $10 to $1,000, $10 to $25, and $4 to $30, respectively, on the black market [50]. Credentials could be used directly by malware authors for greater financial gain, but financial fraud can be difficult to perpetrate and requires specialization [50].

Legitimate premium-rate phone calls and Short Messaging Service (SMS) messages deliver valuable content, such as stock quotes, technical support, or adult services. The cost of a premium-rate call or SMS is charged to the sender's phone bill. Premium rate calls can cost several dollars per minute, and premium rate SMS messages can cost several dollars per message. Premium-rate calls were abused by desktop malware for financial gain in the 1990s and early 2000s, when computers were connected to dial-up modems. Premium-rate SMS messages are stealthier than premium-rate calls because calls tie up the phone line. In Android, malware can completely hide premium-rate SMS messages from the user. Premium-rate SMS attacks could feasibly go unnoticed untilthe user's next phone bill.

Twenty-four of 46 pieces of recent mobile malware send premium rate SMS messages. For example, an application purporting to be a Russian adult video player sent premium-rate SMS to an adult service [51]. Another piece of malware, Geinimi, was set up to send premium SMS messages to number specified by remote commands [52]. Two of 46 malicious applications place premium-rate phone calls. Each of these pieces of malware targets either Android or Symbian devices.

SMS spam has been used for commercial advertising and spreading phishing links. Commercial spammers are incentivized to use malware to send SMS spam because sending SMS spam is illegal in most countries. Sending spam from a compromised machine reduces the risk to the spammer because it obscures the provenance of the spam. Users might not notice the outgoing SMS messages until their monthly phone bills arrive. Even then, users with unlimited SMS messaging plans may never notice the extra

SMS messages. Furthermore, the use of SMS may lend more authenticity to spam than e-mail because phone contacts are often more intimately acquainted than e-mail contacts.

Many web sites rely on search engines for traffic, which makes web site owners desire high visibility in search engine results. Search engines rank web sites according to how relevant each web site is to a given search term. An engine's perception of relevance is influenced by the rate at which users click on the web sites returned for a search term. A web site will rise in the results for a search term if many people search for that term and click on that web site. Malware can be employed to improve a web site's ranking on search engine results. This type of malware sends web requests to the search engine for the target search term. The malware then fraudulently "clicks" on the search result that corresponds to the target web site. As a result, the website's rank for that search term will increase. The value of fraudulent search engine optimization depends on how well the target site can capitalize on its increased visibility, but search engine optimization is a large and lucrative market. One recent Android Trojan, ADRD/HongTouTou, performs search engine optimization. ADRD was built to boost the Baidu search result ranking of a Chinese web site [53]. Desktop malware has also been known to fraudulently perform search engine optimization.

Advertisers may be advertising networks when users view or click on advertisements. In turn, advertising networks pay the web sites that host advertisements. Networks may also be chained in a series, with each network relaying the advertisement and paying the next one in the series. Unscrupulous web sites and advertising networks defraud advertisers and non-malicious networks by using desktop malware to load and click on advertisements [54, 55]. If undetected, click fraud generates a few cents (or even dollars) per instance of fraud. The attacker will directly benefit from the fraud by receiving some portion of the fraudulent payment. An attacker might also launch a click fraud attack on advertising competitors. This depletes the competitors' advertisement budgets, resulting in more legitimate traffic to the attacker's ads. Furthermore, competitors may lower their advertising bids after seeing a lower return on investment, causing the cost of advertisements to go down [56]. Advertising click fraud is very similar to search engine optimization fraud. Although we are not aware of any mobile malware in the wild that performs advertising click fraud, we expect to see it soon.

Many legitimate applications use advertisements to earn money while providing the application to users for free. However, malicious applications can take advertising a step further with invasive advertising practices. Rather than placing advertisements alongside legitimate application content, malicious adware will display advertisements when the user is interacting with other applications. This could significantly interfere with a user's experience with other applications. There are two main reasons for an attacker to display advertisements with malware. First, the attacker may want to advertise goods or services that are illegal or of a nature that legitimate advertising companies prohibit (e.g. pornography, gambling, endangered species products [57, 58]).An attacker might do this to advertise his own products or to create a black market advertising network for affiliates' products. Second, the attacker may simply want to collect revenue from displaying legitimate advertisements. The attacker may be able to generate more revenue with invasive advertising practices by displaying advertisements to users more often or in such a way that users accidentally click on them. This is not considered click fraud because it capitalizes on users' legitimate (albeit accidental) clicks instead of automated clicks. However, these invasive advertising practices are against legitimate networks' terms of service.

Android and iOS support in-application billing, which allows a user to purchase a virtual item from an application using the payment account associated with the Android Market or Apple App Store. Users can therefore buy items such as game credits and music from applications without directly providing the application with payment information. With its growing popularity, in-application billing could be a possible target for the future. First, the implementations of in-application billing protocols could include bugs that allow malware to charge users for items without their approval. Second, malicious applications could use social engineering, click jacking, or phishing attacks to trick users into accidentally or unknowingly approving in-application purchases. For example, iOS sometimes prompts users to enter their App Store passwords into windows that hover over applications, as part of the in-application billing process; these windows are a potential target for phishing attacks. Governments may use mobile phones to monitor citizens and their activities. Unlike the majority of other incentives discussed in this paper, government spying is not motivated directly by financial gain. This type of monitoring could be performed on a large scale (e.g., China's Internet monitoring) or targeted at known dissidents or suspected criminals. It could incorporate GPS tracking, audio and video recording, monitoring of e-mail and SMS messages, and extracting lists of contacts. For example, in 2009 an Internet Service Provider (ISP) in the United Arab Emirates pushed an “update" to 145,000 Android users that drained phone batteries and forwarded copies of e-mails to a government server [59]. Similarly, China partnered with

eBay to produce and distribute a customized version of Skype that censors and tracks users; standard Skype is not available for download in China [60].

The government threat model is significantly more powerful than the criminal threat model. Governments have the ability to gain the cooperation of network carriers and device manufacturers to manipulate firmware updates, control financial markets, and distribute root kits on all devices. Permissions, signing, and anti-virus software can be circumvented by a government that can corrupt the integrity of the operating system. Markets and review processes cannot be trusted to filter out government-sponsored spyware because governments can compel the responsible agencies to publish it. Government agents also can gain physical access to targets' phones to install monitoring software. Unlike criminals, there is little motivation for a government to use arbitrary third party applications to spread malware, unless the government was unable to gain the cooperation of the necessary corporate parties to launch a stronger attack. Governments are more likely to subvert smart phone operating systems or modify specific popular applications (with the original application no longer available).

To perpetrate distributed Denial of Service (DDoS) attacks, botnet owners command large groups of compromised machines to simultaneously send requests to servers. DDoS attacks can be launched for ransom, amusement, cyberwarfare, or as a paid service to others. Traditional DDoS attacks are difficult to stop because of their distributed nature, but one possible approach is for the server to block the IP addresses of visitors that behave anomalously. Consequently, each attacking machine is limited to a smaller number of fraudulent requests. This would not be an effective defense mechanism against mobile-based DDoS attacks because cellular networks assign new IP addresses as often as every few minutes [61]. If that rate of IP assignment is not fast enough, mobile malware can force the assignment of a new IP address from the cellular network by resetting the data connection [61]. In comparison, many desktop machines have static or infrequently-changing public IP addresses that cannot be forcibly reassigned. Despite this advantage, mobile phones also present challenges for DDoS attackers. Mobile phones on cellular networks have significantly less bandwidth than non-mobile devices. Furthermore, an attacker would need to avoid draining the phone's battery too much, limiting a mobile device to a few Hypertext Transmission Protocol (HTTP) requests every few minutes. We expect to eventually see some DDoS malware for mobile phones, but not until phones' bandwidth and batteries are improved.

Apart from DDoS attacks, attackers nowadays are finding ways to perpetrate Near Field Communication (NFC) in large scale. Mobile phones are beginning to incorporate Near FieldCommunication (NFC), which allows short, paired transactions with other NFC-enabled devices in close proximity. NFC can be used for commerce (i.e., accepting credit card transactions), social networking (e.g., sharing contact information), device configuration (e.g., automatically configuring WiFi), and more. It is predicted that mobile payments using NFC will reach $670 billion by 2015 [62].

With growing threat to consumers of the Android powered devices**,** many security agencies and researchers have been working to improve and evolve the detection and prevention of malware attacks. Detection approaches was something that was lacking from the very first release of the Android OS. This had a lot to do with the lack of research into the area. Detection approaches is an invaluable tool to combat malware attacks**,** a lesson learned from the malware wars on PCs. Smartphones are not so different from traditional desktop PC’s nowadays, plenty of applications that were performed on PC can now be performed on smartphones. So they can also suffer from the same weaknesses and vulnerabilities as PC’s. It is foresaid that many of smartphone malware will follow the same path as PC malware as most of technologies needed still exist with malware for PC’s [37]

Also, with the role that the Google powered app store for mobile devices, a place where consumers rarely even conceived an idea of having malware associated with it. With the majority of downloads of mobile applications coming from an openly available and centralized app store, a new security concern and possible vulnerability has arisen that hasn’t been widely experienced in the PC world. This possible vulnerability ultimately exposes the great majority to of Android users to an easy avenue for malware developers to attack.

This survey will serve to inform the reader of such topics related to mobile security, using trends and the evolution of the Android mobile platform to cover the topics. The survey is intended for all with an interest in mobile security in general and with minimal background or knowledge of the subject.

2. BRIEF HISTORY OF ANDROID

An open source Linux based operating system, Android was purchased by Google in 2005[16]. Android was founded with the Open Handset Alliance, and finally released for mobile devices such as smartphones and sold its first smartphone in 2008. Android was also designed to make it

easy for developers to program the device in languages such as C, C++, and Java. Google also provides a freely available software development kit (SDK) to facilitate application creation. Android powered devices have grown to be a common sight internationally today, leading the global smartphone marketplace share for mobile operating systems as of early 2013 at approximately 70% [19]. Android has seen numerous updates spanning from version 1 through 4.2.x providing new features, performance boost, design changes, as well security patches.

While Google had means for offering mobile apps from almost the beginning, it wasn't until March of 2012 that they transitioned to their current app store, the Google Play store [20]. As of early 2013 the primary app store for Android Google Play has accumulated over 700,000 apps according to its website, easily making it one of the largest mobile app stores in the Industry. In late 2012 it was reported that approximately 25 billion downloads were made from Google Play [21], not considering for all apps downloaded from numerous other unofficial app stores and various sources.

3. EVOLUTION AND GROWTH OF ANDROID MOBILE MALWARE

With the enormous popularity and growth of the Android platform has seen since its inception, it not surprising that it's become a more lucrative target for malware designers. The Android platform is designed to allow developers to use core device functionality such as the text messages and the calling features [14]. The Android platform debuted on only 1 phone on one carrier and now is offered on hundreds of phone across every major carrier. In recent years the number of mobile malware on the Android platform has begun alarming security experts and customers alike. During the 3rd quarter of 2012 a security group F-Secure detected over 51,000 malware instances an increase by 10 folds from the previous 2nd quarter where only approximately 5,000 instances. Among them only 146 were from the Google Play store [22].

The growth and adoption rate for Android has seen a positive increase since its debut and with 2009-quarter estimates from a research company Canalys showing 2.8% market share to a dominating 70% in the first quarter in 2013. Google reported in 2011, that there were 550,000 activations daily and growing by approximately 4.4% per week [23]. It was these kinds of number that attracted such a large malware developing community for PCs. Android today can be seen used in international communities such as South America and China even though China has had limited access to Google services including the Google Play store.

The evolution of the Android platform has seen several version changes from 1.x when first revealed in 2007 and now its latest iterations as of early 2013 codename Jelly Bean version 4.x.x. Each version has added new features and boasted overall performance as well as closing security holes and resolving vulnerabilities. Unfortunately, a slow adaption to the latest versions has meant that many of these vulnerabilities have remained throughout the updates. A sample was taken using data from Google's Play store to get a representative measure on the distribution of different currently being used.

Below is a chart of Android version distribution measured in April of 2013 [15].

Version	Codename	API	Distribution
1.6	Donut	4	0.1%
2.1	Eclair	7	1.4%
2.2	Froyo	8	3.1%
2.3.3-2.3.7	Gingerbread	10	34.1%
3.2	Honeycomb	13	0.1%
4.03-4.04	Ice Cream Sandwich	15	23.3%
4.1.x	Jelly Bean	16	32.3%
4.2		17	5.6%

Android Version Distribution

With the explosive growth and popularity in Android mobile devices it has become very clear that mobile security has been an ever more important topic. Between August of 2010 and October of 2011, researchers were able to collect more than 1,200 malware samples covering the majority of existing Android malware families [2], and the researchers evaluated mobile security software. Experimentally, it was

found that in the best cases 79.6% of the malware in the dataset were detected while in the worst cases only 20.2% were successful detected. The research finding clearly demonstrated the need to improve anti-mobile-malware solutions. The research data also pointed out that the majority of the malware were not from the official Android Market but from alternative sources. The major categories the researchers used to classify the different malware included Privilege Escalation, Remote Control, Financial Charges, Personal Information Stealing. Many of the malware collected fit into more than one category.

The evolution of some of these malwares seems to be progressing rapidly. For example the DroidKungFu malware, which was initially, detected in the summer of 2011 and by the 4th quarter, researchers have found 4 other versions [2]. As of time of their publication a total of 473 DroidKungFu variants samples were obtained. This demonstrates the rapid development and evolution of malware. With growing phone capabilities the kinds of malware to look for is also evolving. A proof of concept malware named StuxMob was created to demonstrate situational-aware malware for targeted attacks [8]. The possibility to target mobile devices based on profiles and users opens the doors to whole new kind of attacks. Profiles based on readings from the devices available sensor allowing, for example the phone's ability to know when a user is performing certain activities such as running, walking, or even seating down at work. As the capabilities of these Android powered devices continue to grow, so does the capability of malware.

4. DETECTION APPROACHES

When Android OS entered the market in late 2008, detection of malware approaches that were used for Android operating system were insufficient. A considerable amount of work has been made in the area of malware detection. Several approaches as in [39], [40], [41] monitor power usage of applications and reports an anomaly in consumption. Other techniques [42], [43] use system call monitoring to detect unusual system call patterns. There has been significant work on the problem of detecting malware on mobile devices. Several approaches [39], [40], [41] monitor the power usage of applications and report an anomalous consumption. Others [42], [43] monitor system calls and attempt to detect unusual system call patterns, use more traditional comparison with known malware (e.g. [44]) or other heuristics (e.g. [45]).

The more general field of malware detection is hosted to a wider range of approaches. Traditional static analysis approaches such as [38], [46], which focus on comparing programs with known malware based on the program code, looking for signatures using other heuristics. Other approaches [47], [48], [49] focus on using machine learning and data mining approaches for malware detection. In [49], Tesauro et al. train a neural network to detect boot sector viruses, based on bytestring trigrams. Schultz et al. [48] compare three machine learning algorithms trained on three features: DLL and system calls made by the program, strings found in the program binary, and a raw hexadecimal representation of the binary. In [47], Kolter and Maloof train several machine learning algorithms on byte string n-grams.

The early prototypes of Android malware detection were insufficient due to the lack of malware samples for Android OS. In those early times, many users of Android were developers and tech hobbyists, so they were knowledgeable about cybersecurity and risks that were associated with it. Because of that, early Android infrastructure did not have advance security built-in. The reason of that, Android developers were aiming for their operating system to be compatible with existing code.

Since then, Android project have been enhanced by diverse developer's ideas that focus on improving: smartphone computational capability, high-speed mobile communication network, adapting new technological advances, and invitations. From Android early development, users had access to the Google App store, which is now called Google Play. Developers publish applications (commonly called apps) for their customers through Google Play. Some of these applications are paid, which contribute to the global market.

However, having an online App store for a smartphone was not a new idea, but allowing anyone to publish without code evaluation was something unique. This idea is the core of the open-source movement, and it encourages developers to contribute in Android development and populate Google Play with applications. Due to the collaborative work that initiated Android, the operating system is shipped for free. That leads to a smartphone that is powered by Android to become increasingly inexpensive and more popular than its competitors. Google Inc. embraces openness to its App Store, so any developer can upload the application, and also that developer can profit from it if he or she chooses.

In the early years of Android development, Google introduced an App store (now called Google Play). Although the store was supposedly inherently safer, there are already several cases that show that Google Play store is not free from malware. Its vulnerability can threaten the end-users, and even allow identity theft, which can lead to serious consequences due to open source philosophy.

However, the unsustainable growth of Android phone activations led to the number of malware samples increasing exponentially [11]. Developers and organizations started collecting sufficient samples of Android malwares. Some of them shared interesting findings after analyzing those samples. There are several proposed ideas to detect malware in Google Play store or in the smartphone through malware detection.

4.1 MALWARE DETECTION IN SMARTPHONE

Android OS has full proper operating system functionality because it uses Linux kernel, GNU's Not Unix (GNU) tool chain, and other existing tools in its infrastructure. The capabilities of smartphones powered by Android can compete with traditional personal computers, but there are some drawbacks such as their limited resources. Malware detection theories that have been developed for personal

computer architecture can be used for smartphones powered by Android because its infrastructure supports upward-compatibility.

However, many developers are avoiding the use of a traditional malware detection theory due to the finite phone resources. On personal computer architecture, performing exhaustive searches, which requires huge computational power, is not a major problem. What is worth a mention is that usually computational power has a direct relationship with power-consumption. For example, computational power has been increased due to application needs, but that also means the application is consuming more battery power. From a PC point of view, if the malware detection software consumes a lot of power, which is fine because PCs are not meant to be as portable or mobile and are connected to an outlet instead of a battery. That is why almost all malware detection that designs for a personal computer uses an exhaustive search to detect hosted malwares in the system. The reason for this detection design adaptation is, despite its huge power-consumption, detection algorithm works effectively.

On the contrary, an efficient detection algorithm and an efficient battery usage are a must for a mobile phone. That is why many developers avoid importing an existing work from a PC directly to an Android smartphone. The phone battery would drain out quickly by just performing an exhaustive search to detect malware because the detection method is not applicable to lower computing capability and power-limited smartphones. Therefore, a malware detection mechanism with an efficient and low battery usage is desirable for a smartphone powered by Android.

Before summarizing other people's works, there is a common detail that most of the papers address. When finite phone resources and more exhaustive monitoring capability create a higher demand on the device, draining the battery occurs much faster than expected. However, many approaches analyze the low system information, which require a complicated sorting and string searching. For example, there are Android malwares contain these function names:SendTextMessage(),SendMultipartTextMessage(),get PhoneService, and getCurrentLocation(). In fact, these strings are the most used SDK functions by malware, so by performing a string search the detector can spot these malicious programs. Therefore, the objective of performing malware detection in the Android smartphone is an algorithm optimization that may provide a less computation time which would reduce the power-consumption.

There are many attempts to provide an efficient detection algorithm and an effective battery usage for Android smartphones. For example, Forrest has presented typical host-based anomaly detection for Android smartphone that takes power-consumption into consideration [11]. Host-based anomaly detection is a way of monitoring system call sequence stored in the database. For example, if a program behavior has not appeared in a system call sequence database, the detector would consider the program as a malicious program. After the detector has spotted the malicious program, the detector will inform the system to do the necessary processes of isolation the malicious program [12].

Forrest also enhanced and developed his malware detector by adding these features: behavioral learning algorithms, finite state machines, and hidden Markov chain methods. However, despite of these improvements Forrest's malware detector lacks the existing semantics of system calls which can allow some malware to escape the detection. There should be a runtime trace of application behavior in Android framework to overcome Forrest's malware detector limitation.

Forrest is not the only one who developed malware detection for Android smartphones. In fact, most of the developed malware detectors are using a similar algorithm. Forrest's detector algorithm detects malware by comparing program's behavior with any malicious activity that malware most likely would perform. In other words, if the detector has spotted a number of unusual system calls, then the detector would label the program as malicious. However, there are some malwares that can detect the presence of monitoring mechanism (or software) in the phone, so the malware would not perform any malicious activities when the detector is active. That is when Forrest's malware detector starts missing some existing malwares because these malwares would stop doing any malicious activities in the device when the detection process is in action.

In [27], a malware detector framework is proposed based on permissions of Android applications. This framework uses machine-learning techniques to make a decision on whether a current application is malware or not. A different machine-learning framework, Crowdroid [28] is used that recognizes Trojan-like malware on Android smartphones. This scheme analyzes the number of system calls issued by a particular application during the execution of an action requiring user interaction. A trojanized application can be detected by observing the difference in type and number of times a system call is issued. Another example of IDS that relies on machine learning techniques is Andromaly [29] which observes several parameters monitoring both the smartphone and user's behaviors, spanning from sensors activities to CPU usage. In this work, 88 features were used to describe observed behaviors, which are further pre-processed using feature selection algorithms. The authors developed four malicious applications in order to evaluate the ability that aided detection of anomalies. [30] described a global malware detection approach, MADAM: Multi-Level Anomaly Detector for Android Malware that is capable of detecting malware contained in unknown applications. This detector uses 13 features to detect malware for both kernel level and user level. [30] includes framework that consist of a monitoring client, Remote Anomaly Detection System (RADS) and a visualization component in order to monitor smartphones to extract features that can be used in a machine learning algorithm to detect anomalies. A behavior-based malware detection system (pBMDS) is proposed in [31] that use correlation between user's inputs and system calls in order to detect anomalous activities related to SMS/MMS sending. A new service named Kirin security service for Android is described in [33] and [34] that perform lightweight certification of applications to mitigate malware at install time. This service uses security rules, which matches undesirable properties in security configuration, bundled with applications. In [35], a static analysis on the

executable to extract functions calls usage is described that uses readelf command. Lastly, in [36], some security solutions for mobile devices are explained.

Zhao, Zhang, Ge, and Yuan have proposed a solution to monitor system calls in runtime without malwares having the ability of noticing the detector presences [11]. They developed an application for Android smartphone called RobotDroid, which is a software behavior signature, based on malware detection framework. What is noticeably different from previous approaches, RobotDroid uses an active learning algorithm [12]. The previous approaches have used a passive learning algorithm, which means after collecting the data the detector would perform the necessary analysis, but that makes the detector vulnerable for malware deceptions, because the hackers can update their malware to deceive the passive learning algorithm which is used in the malware detector.

To make sure the detector would not exclude any malware from the collected information, the active algorithm would stop any unusual activities, and then to record its existence. The malware RobotDroid is powered by SVM Active learning algorithm, which is an efficient solver for collected information in a runtime [12].

As a result of enhancing the malware detector, RobotDroid can perform a variety of functions: detect broader range of malicious software, analyze system calls in runtime, and can extend its malware characteristics database dynamically. With all of these functions and features, RobotDroid is still able to use system power wisely. Zhao, Zhang, Ge, and Yuan have tested their detector RobotDroid and experimental results show that their method has a good applicability and scalability. In fact, RobotDroid can detect a variety of popular known as well as unknown malware. It seems that monitoring software behavioral activity in Android framework is an accurate technique to determine the behavior of Android applications. By utilizing what the Android system can provide which is detailed and effected low level information, but the detection algorithm must be optimized.

Algorithm optimization to reduce a power-consumption has addresses new challenges that RobotDroid fulfilled in some categories and failed in others. The developers of RobotDroid have archived an active detection, which is powered by an active learning method and developing dynamic database.

On the other hand, RobotDroid lacks some features. For example, the system would always separate the software behavioral signature vectors in two sets even if there is no malware on it. The reason of that is when RobotDroid detects a malicious signature enters into the normal dataset. RobotDroid would most likely have inaccurate signature sequence mapping for system calls that are detected in the system. This issue can lead up to infinite replication, so it would require some manual check or further automatic analysis. In short, to overcome this problem, RobotDroid must always separate the software behavioral signature vectors in two sets even if there is no malware on it. In other words, these two sets are used to cancel the replication of system calls.

Shabtai has also proposed a detector that has two sets [23]. His malware detector spots suspicious temporal patterns as malicious behavior. These suspicious temporal patterns are known as knowledge-based temporal abstraction. These abstractions can be information theft, power exhaust, and botnet. However, Shabtai's detection is far from perfection. The detector does not secure the user IP. This feature has been excluded due to algorithm optimization, but the user device is vulnerable to an attacker. To overcome this problem, Shabtai encrypted whole phone information, which increased the computational power, and not just for the detector, but to the whole phone applications. In the end, his detector algorithms are optimized, but the Android network security was not enough to protect the user information, so he encrypted the whole phone information to be secure. By encrypting the whole phone information would definitely drain the phone faster.

Burguera and Zurutuza go deeper than just analyzing system calls in the user mode; they did their analysis in the kernel mode [24]. By monitoring system call in Android kernel level, the system can provide a full control of any system call. That means having a better Android security can be achieved in the kernel mode. However, in the kernel mode doing a mistake can turn the system down because the kernel can preempt any process in the system. By using the kernel mode, they can generate software behavioral patterns and classify these patterns by using cluster algorithms. Their approach is successful, but it requires the user to know advance topics such as debugging the kernel.

In summary, all the approaches that have been examined in this section are developed for detecting continuous attacks. There is still long way to reach the optimal Android malware detection because most of this software does not have a user-friendly interface. So, even by achieving the optimal malware detection for smartphones powered by Android have accurate results. There are still long way to make it usable for the massive distribution.

4.2 MALWARE DETECTION IN APP STORE

Most of these trojanized applications use SDK function call executions. Google provides the SDK to help the developers avoid Android fragmentation, but standardized API gives hackers much broader users to attack.

Most users trust Google Play and some also trust third-party App stores, but the landscape has been changed. The user can be affected by cyber-criminality even when downloading their Apps from legitimate store such as Google Play. There was a study from TrendLabs engineers showing that the user still most likely gets affected by trojanized applications by using a legitimate store. This pitfall comes from the idea of Android embraces openness.

Needless to say, this type of ecosystem increases productivity rapidly, so developers start coding and collaborating in every software category. As an open source mobile operating system promises commitment to openness and opportunity to everybody, developers rushed, and then

consumers followed. The number of android activations has skyrocketed.

That is why many big name corporations and industries, such as the banking sector, start to publish their application in Google Play store for the sake of customers' convenience.

Anyone can contribute and upload their applications; also someone can even download an existing application, do some modifications into the downloaded code, and upload the modified version to Google Play. This is the core of open source progress and development, but hackers can use this protocol for their selfish means. Example, a hacker downloads a banking application, inserts malicious codes, and then that hacker publishes the malicious application into Google Play store as the bank to deceive end-users [10].

The trojanized application can infect and victimize a sheer number of end-users. That is why there are many ideas and invitations to decrease the overwhelming number of victims by exploiting the malicious applications without jeopardizing the idea of Google Play openness. For example, a malware that was in Google Play store, named DroidDreamLight. This particular malware has since been taken down by Google from Google Play and has been deleted from user's phones, but only after many users fell prey. In fact, DroidDreamLight has affected 30,000-120,000 users in May 2011 by stealing their information and sending this stolen information to cybercriminals.

What is clear from the gathered facts, most malware repackaging happens for popular applications. End-users get victimized because the applications are desirable to execute. Most users do not check if the application that they want to install does not have a replica [13]. And many fail to check the installation link was emailed to them. Machine learning techniques have been widely applied for classifying applications mainly focused on generic malware detection [1-5]. These classifications can detect repackaged programs in an App store. There are several approaches that have been proposed to try to classify applications specifying the malware class, which are: Trojan, worms, virus, and other malware types. For example, Shabtai trained machine-learning models using as features the count of elements, attributes or namespaces of the parsed apk [8]. To evaluate their models, they selected features using three selection methods: Information Gain, Fisher Score and Chi-Square. They obtained 89% of accuracy classifying applications into only 2 categories: tools or games.

A method for classifying Android applications using machine-learning techniques can reduce the number of victims that download a trojanized program. Other example, Sanz, Santos, and Laorden proposed an automatic categorization of Android applications [11]. The main concept is to provide an automatically characterization for different types of applications. By performing automatic sorting for the App store's applications, the sorting can be empowered by detection mechanism to exploit malicious applications. The detection mechanism method that they use is a machine-learning technique to represent each application to different feature sets, which are:

1. the frequency of occurrence of the printable strings
2. the different permissions of the application itself
3. the permissions of the application extracted from the Google Play

Vidas and Christin observed that repackaged programs can be detected from the App store without restricting how developers publish their programs. The way to a secure App market is by performing a verification protocol that they proposed [13]. They called their method "AppIntegrity," which they claims a proof-of-concept implementation. Applications can be authenticated that are offered in an App store such as Google Play. The authentication process can make it difficult for a repackaged application to reenter the App store. Their aim is to perform the minimum computational or communication overhead as possible.

AppIntegrity uses an end-to-end verification protocol to reduce the threat of repackaging. Both endpoints developers and consumers have an encryption key, and so the information that would propagate through the communication channel is encrypted. A communication channel encryption does not allow a malicious patch to attach itself to the program. The encryption method is commonly used for personal computers, but because it does not consume a lot of power, it can be used for Android smartphone. The protocol has been tested on PC's and Android devices, but AppIntegrity can be used to other application markets.

Because AppIntegrity uses an end-to-end verification protocol, the implementation cost is reasonable. Ideally, AppIntegrity just needs a minimal network and local resource use for constraining a mobile device's environment. The environment can access Google Play or other App stores. In fact, AppIntegrity requires no changes to the existing Android development process. Minimal changes to the Android framework could enhance the ability for protection of AppIntegrity users, but even when used with the current version of Android, AppIntegrity can provide added safety by rapidly uninstalling unverified applications, and providing building blocks for future protocols and services.

5. AN APP STORE ACCESSIBLE TO GLOBAL CUSTOMERS

The most popular and widely used Android app store for both consumers and developers alike, Google Play store has been greatly considered to be the safest place to acquire new apps. It's been a strong belief that users are safe as long as their apps are all signed apps from the Google Play store and it's a belief that has held well when looking at the data from researchers. But a recent report outlined a critical flaw affecting all versions of Android devices vulnerable to hackers looking to get full control over your device [25]. The flaw allows developers to insert code into digitally signed apps, which includes all the apps in the Google Play store, and allows them to be turned into potential malware.

These digital signatures are what have been used in the past to differentiate between safe and unsafe mobile apps for Android. Currently, the largest market for Android devices is in China and it also accounts for the largest percentage of malware attacks [19]. China actually has an un-proportionally high number of malware attacks compared with other markets like the United States and Canada. One of the most notable differences between the US and Chinese markets are the restrictions on using the official Google Play store, forcing many in China to use alternative sources and even questionable pirated apps which are not digitally signed. It seems that the majority of apps in China are downloaded from Chinese app stores or pirate sites [24]. This is an observation that helps Google's argument regarding the Play store being relative safe and not a major security risk for consumers. But now with the vulnerability that affects all signed apps discovered, and no solution yet, Google Play customers can realistically see a sharp rise in malware attacks to rival the numbers seen in China.

CONCLUSIONS

Mobile malware are at a rise including those that pose a threat to Android users. With the growing potential to cause greater harm to its victims, the security threat must be answered with an ever more aware community as the number of malware seems to only be increasing at an ever faster rate. The topics discussed clearly demonstrate the existence of the threat and its growing numbers as well as some of the existing efforts in response to this threat. While the solution to completely curve the threat level cannot be derived from this survey, it still serves to inform its audience of the threat by bringing forth key topics. Understanding the threat and current trends can help to predict to some degree the level of danger malware will pose in the near future to Android users.

REFERENCES

[1] M. La Polla et al., "A Survey on Security for Mobile Devices," in IEEE Communications Surveys & Tutorials, VOL. 15, NO. 1, First Quarter 2013

[2] Y. Zhou and X. Jiang, "Dissecting Android Malware: Characterization and Evolution," in IEEE Symposium on Security and Privacy, 2012, p.95-109

[3] J. Li et al., "Android Malware Forensics: Reconstruction of Malicious Events," in 32nd International Conference on Distributed Computing Systems Workshops, 2012, p. 552-558

[4] D. Wu et al., "DroidMat: Android Malware Detection through Manifest and API Calls Tracing," in Seventh Asia Joint Conference on Information Security, 2012p.62-69

[5] J. Sahs and Latifur Khan, "A Machine Learning Approach to Android Malware Detection," in European Intelligence and Security Informatics Conference, 2012, p. 141-147

[6] A. P. Felt et al., "A Survey of Mobile Malware in the Wild," Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices, 2011, Chicago, Illinois, USA

[7] M. Becher et al., " Mobile Security Catching Up? Revealing the Nuts and Bolts of the Security of Mobile Devices," in IEEE Symposium on Security and Privacy, p.96-111, 2011 [doi>10.1109/SP.2011.29]

[8] S. Zawoad et al., (2013) "*Poster: StuxMob: A Situational-Aware Malware for Targeted Attack on Smart Mobile Devices* ," Available: http://www.ieee-security.org/TC/SP2013/posters/Shams_Zawoad.pdf

[9] M. Herpich et al., "A trusted ecosystem for Android applications based on context-aware access control" in Malicious and Unwanted Software (Malware), 2012, pp 73 - 78

[10] From DroidDreamLight lurks Behind Legitimate Android Apps:

[11] B. Sanz, I. Santos, C. Laorden, "On the Automatic Categorisation of Android Applications" in The 9th Annual IEEE Consumer Communications and Networking Conference - Security and Content Protection, 2012

[12] M. Zhao , T. Zhang , F. Ge , and Z. Yuan , " RobotDroid: A Lightweight Malware DetectionFramework on Smartphones ," JOURNAL OF NETWORKS, VOL. 7, NO. 4, APRIL 2012

[13] T. Vidas and N. Christin, "Sweetening Android Lemon Markets: Measuring and Combating Malware in Application Marketplaces ," ACM, San Antonio, Texas, USA February 18–20, 2013

[14] Android Overview [Online] Available:http://www.openhandsetalliance.com/android_overview.html

[15] Dashboard[Online] Available:http://developer.android.com/about/dashboards/index.html

[16] Elgin, Ben "Google Buys Android For Its Online Arsenal" August 2005[Online] Available:http://www.webcitation.org/5wk7sIvVb

[17] Android Overview[Online] Available:http://www.openhandsetalliance.com/android_overview.html

[18] "Announcing the Android 1.0 SDK" September 2008[Online] Available:http://android-developers.blogspot.in/2008/09/announcing-android-10-sdk-release-1.html

[19] Lunden, Ingrid "Android, Led By Samsung, Continues To Storm The Smartphone Market, Pushing A Global 70% Market Share " July 2013 [Online] Available: http://techcrunch.com/2013/07/01/android-led-by-samsung-continues-to-storm-the-smartphone-market-pushing-a-global-70-market-share/?ncid=tcdaily

[20] Wimberly, Taylor "Android Market is dead, Google Play takes over starting today" March 2012 [Online] Available: http://androidandme.com/2012/03/news/android-market-is-dead-google-play-takes-over/

[21] Rosenberg, Jamie "Google Play Hits 25 Million Downloads" September 2012[Online] Available: http://officialandroid.blogspot.ca/2012/09/google-play-hits-25-billion-downloads.html

[22] H, Michael, "Android malware perspective: only 0.5% comes from the Play Store" November 2012[Online] Available: http://www.phonearena.com/news/Android-malware-perspective-only-0.5-comes-from-the-Play-Store_id36696

[23] Kumparak, Greg "Android Now Seeing 550,000 Activations Per Day" July 2011[Online] Available: http://techcrunch.com/2011/07/14/android-now-seeing-550000-activations-per-day/

[24] Armasu, Lucien "Wind-up Kinght developer: Piracy rates on iOS and Android are comparable, China is the main source"July 2012 [Online]Available: http://www.androidauthority.com/piracy-rates-are-higher-ios-android-wind-up-knight-developer-104305/

[25] SANS NewsBites - Volume: XV, Issue: 53 [Online] Available: http://www.sans.org/newsletters/newsbites/newsbites.php?vol=15&issue=53&rss=Y#sID201 Available: http://www.androidauthority.com/piracy-rates-are-higher-ios-android-wind-up-knight-developer-104305/

[26] SANS NewsBites - Volume: XV, Issue: 53 [Online] Available: http://www.sans.org/newsletters/newsbites/newsbites.php?vol=15&issue=53&rss=Y#sID201

[27] Zarni Aung and Win Zaw, "Permission-Based Android Malware Detection," INTERNATIONAL JOURNAL OF SCIENTIFIC & TECHNOLOGY RESEARCH VOLUME 2, ISSUE 3, MARCH 2013

[28] I. Burguera, U.Z., Nadijm-Tehrani, S.: Crowdroid: Behavior- Based Malware Detection System for Android. In: SPSM'11, ACM(October 2011)

[29] A. Shabtai, U. Kanonov, Y. Elovici, C. Glezer, Y. Weiss: Andromaly: a behavioral malware detection framework for android devices. Journal of Intelligent Information Systems 38(1) (January 2011) 161-190

[30] G. Dini, F.Martinelli, A. Saracino, D. Sgandurra: MADAM: a Multi Level Anomaly Detector for Android Malware

[31] Schmidt, A.D., Peters, F., Lamour, F., Scheel, C., Camtepe, s.A., Albayrak, S.: Monitoring smartphones for anomaly detection. Mob. Netw. Appl. 14(1)(2009) 92-106

[32] Xie,L.,Zhang,X.,Seifert, J.P.,Zhu, S.: pBMDS: a behavior-based malware detection system for cellphone devices. In: Proceedings of the Third ACM Conference on Wireless Network Security, WISEC 2010, Hoboken, New Jersey, USA, March 22-24 2010, ACM(2010) 37-48

[33] Enck, W., Ongtang, M., McDaniel, P.: On lightweight mobile phone application certification. In: CCS '09: Proceedings of the 16th ACM conference on Computer and Communication Security, New York, NY, USA, ACM (2009) 235-245

[34] Ongtang, M., McLaughlin, S., Enck, W., McDaniel, P.: Semantically Rich Application-Centric Security in Android. In: Computer Security Applications Conference, 2009. ACSAC '09. Annual.(Dec 2009) 340-349

[35] Schmidt, A.D., Bye, R., Schmidt, H.G., Clausen, J.H., Kiraz, O., Yuksel, K.A., Camtepe, S.A., Albayrak, S.: Static Analysis of Executables for Collaborative Malware Detection on Android. In Proceedings of IEEE International Conference on Communications, ICC 2009, Dresden, Germany, 14-18 June 2009, IEEE (2009) 1-5

[36] La Polla, M., Martinelli, F., Sgandurra, D.: A survey on security for mobile devices. Communications Surveys Tutorials, IEEE PP(99) (2012) 1-26

[37] A. Schmidt and S. Albayrak, "Malicious software for smartphones," Technische Universität Berlin, DAI-Labor, Technical Paper, vol. 2, pp. 1–53, 2008.

[38] Mihai Christodorescu and Somesh Jha. Static analysis of executables to detect malicious patterns. In Proceedings of the 12th conference on USENIX Security Symposium - Volume 12, SSYM'03, pages 12–12, Berkeley, CA, USA, 2003. USENIX Association.

[39] Bryan Dixon, Yifei Jiang, Abhishek Jaiantilal, and Shivakant Mishra.Location based power analysis to detect malicious code in smartphones.In Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices, SPSM '11, pages 27–32, 2011.

[40] Hahnsang Kim, Joshua Smith, and Kang G. Shin. Detecting energygreedy anomalies and mobile malware variants. In Proceedings of the 6th international conference on Mobile systems, applications, and services, MobiSys '08, pages 239–252, 2008.

[41] Lei Liu, Guanhua Yan, Xinwen Zhang, and Songqing Chen. Virusmeter: Preventing your cellphone from spies. In Proceedings of the 12th International Symposium on Recent Advances in Intrusion Detection, RAID '09, pages 244–264, 2009.

[42] Iker Burguera, Urko Zurutuza, and Simin Nadjm-Tehrani. Crowdroid: behavior-based malware detection system for android. In Proceeding of the 1st ACM workshop on Security and privacy in smartphones and mobile devices, SPSM '11, pages 15–26, 2011.

[43] Liang Xie, Xinwen Zhang, Jean-Pierre Seifert, and Sencun Zhu. pbmds: a behavior-based malware detection system for cellphone devices. In Proceedings of the third ACM conference on Wireless network security, WiSec '10, pages 37–48, 2010.

[44] Abhijit Bose, Xin Hu, Kang G. Shin, and Taejoon Park. Behavioral detection of malware on mobile handsets. In Proceedings of the 6th international conference on Mobile systems, applications, and services, MobiSys '08, pages 225–238, 2008.

[45] Yajin Zhou, Zhi Wang, Wu Zhou, and Xuxian Jiang. Hey, you, get off of my market: Detecting malicious apps in official and alternative android markets. In

Proceedings of the 19th Network and Distributed System Security Symposium, 2012.

[46] Raymond W. Lo, Karl N. Levitt, and Ronald A. Olsson. Mcf: a malicious code filter. Computers & Security, 14(6):541 – 566, 1995.

[47] J. Zico Kolter and Marcus A. Maloof. Learning to detect and classify malicious executables in the wild. J. Mach. Learn. Res., 7:2721–2744, December 2006

[48] Matthew G. Schultz, Eleazar Eskin, Erez Zadok, and Salvatore J. Stolfo.Data mining methods for detection of new malicious executables. In Proceedings of the 2001 IEEE Symposium on Security and Privacy, SP'01, pages 38–, Washington, DC, USA, 2001. IEEE Computer Society.

[49] A. Walenstein, R. Mathur, M.R. Chouchane, and A. Lakhotia. Normalizing metamorphic malware using term rewriting. In Source Code Analysis and Manipulation, 2006. SCAM '06. Sixth IEEE International Workshop on, pages 75–84, 2006.

[50] M. Fossi (Editor). Symantec Report on the Underground Economy. Symantec Corporation, 2008.

[51] Juniper Global Threat Center. Fake player. http://globalthreatcenter.com/?p=1907.

[52] Symantec. Android.geinimi. http://www.symantec.com/security_response/writeup.jsp?docid=2011 010111-5403-99

[53] T. Strazzere. Security Alert: HongTouTou, New Android Trojan, Found in China. The Lookout Blog, 2011.

[54] B. Miller, P. Pearce, C. Grier, C. Kreibich, and V. Paxson. What's Clicking What? Techniques and Innovations of Today's Clickbots. In DIMVA, 2011.

[55] N. Daswani and M. Stoppelman. The anatomy of Clickbot. A. In Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets, pages 11{11. USENIX Association, 2007.

[56] N. Daswani, C. Mysen, V. Rao, S. Weis, K. Gharachorloo, and S. Ghosemajumder. Online advertising fraud. Crimeware: Understanding New Attacks and Defenses, 2008.

[57] Adwords content guidelines. http://adwords.google.com/support/aw/bin/static.py?hl=en&guide=28435&page=guide.cs

[58] Google AdSense Program Policies. https://www.google.com/adsense/support/bin/answer.py?answer=48182

[59] B. Thompson. UAE Blackberry update was spyware. http://news.bbc.co.uk/2/hi/technology/8161190.stm.

[60] J. Marko. Surveillance of Skype Messages Found in China. New York Times, 2008.

[61] M. Balakrishnan, I. Mohomed, and V. Ramasubramanian. Where's That Phone? Geolocating IP Addresses on 3G Networks. In IMC, 2009

[62] M. Calamia. Mobile payments to surge to $670 billion by 2015. http://www.mobiledia.com/news/96900.html , 2011.